Akansha Balhara

- Roll no.: 1/23/SET/BCS/433
- Date : 12/02/2026
- Exp no. : 4

Single Neuron Neural Network

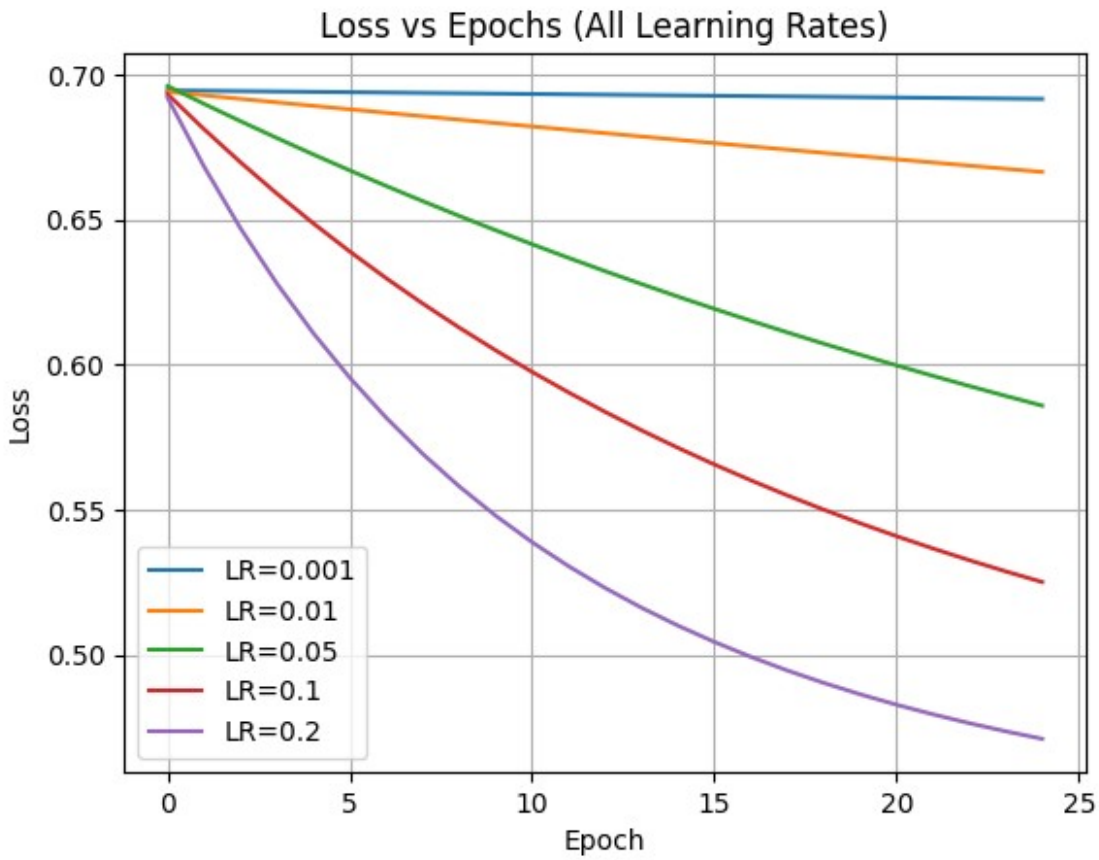
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score

# ========================================
# 1. Load Dataset
# ========================================
df = pd.read_csv("/content/spam_or_not_spam.csv")

print("\n===== FIRST 5 ROWS =====")
print(df.head())

print("\n===== LAST 5 ROWS =====")
print(df.tail())

# Clean data
df = df[['email', 'label']].dropna().reset_index(drop=True)

X = df['email']
y = df['label'].values

# ========================================
# 2. Train-Test Split
# ========================================
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# ========================================
# 3. TF-IDF Vectorization
# ========================================
vectorizer = TfidfVectorizer(
    stop_words='english',
    max_features=4000,
    ngram_range=(1,2)
)

X_train_vec = vectorizer.fit_transform(X_train).toarray()
```

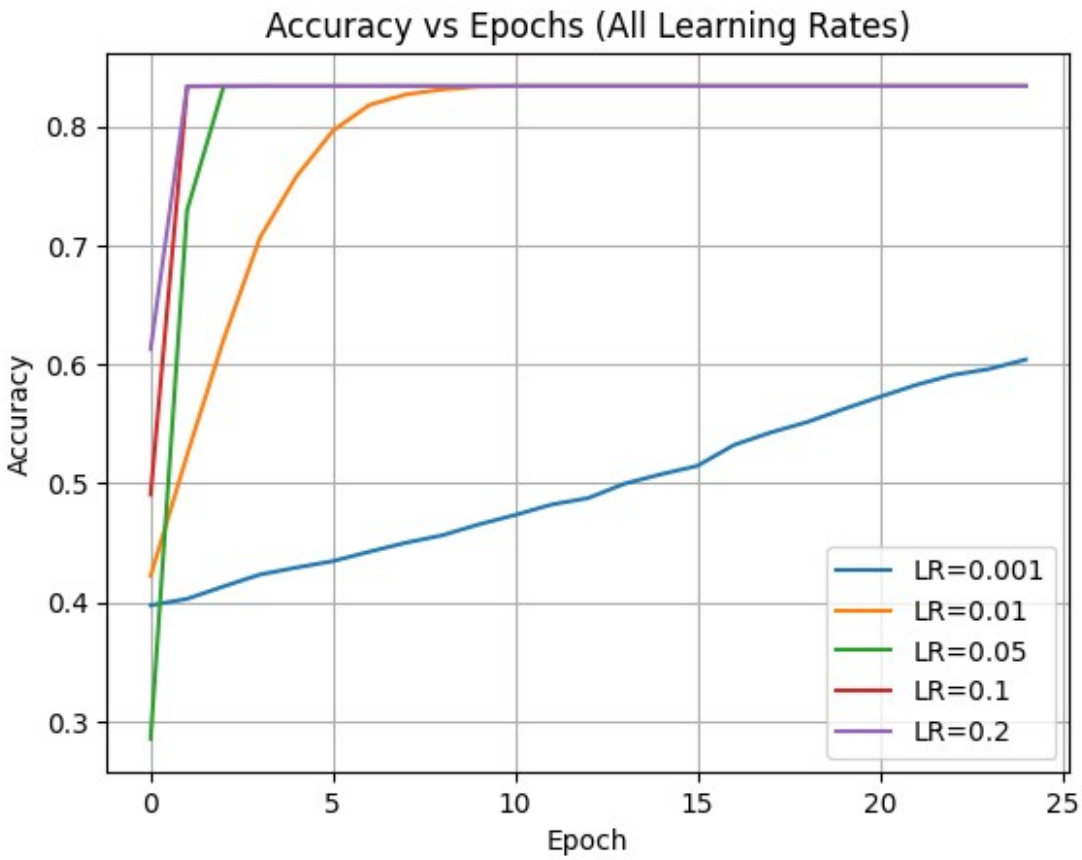
```
X_test_vec = vectorizer.transform(X_test).toarray()

# ========================================
# 4. Sigmoid Function
# ========================================
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

# ========================================
# 5. Training Function
# ========================================
def train_model(learning_rate):

    epochs = 25
    reg = 0.01
    m, n = X_train_vec.shape

    # Random initialization (different results each run)
    W = np.random.randn(n) * 0.01
    b = 0

    loss_history = []
    acc_history = []

    for epoch in range(epochs):

        # Forward pass
        linear = np.dot(X_train_vec, W) + b
        preds = sigmoid(linear)

        # Compute gradients
        error = preds - y_train
        dW = (np.dot(X_train_vec.T, error) / m) + reg * W
        db = np.mean(error)

        # Update parameters
        W -= learning_rate * dW
        b -= learning_rate * db

        # Loss
        loss = -np.mean(
            y_train*np.log(preds + 1e-8) +
            (1 - y_train)*np.log(1 - preds + 1e-8)
        )
        loss_history.append(loss)

        # Training Accuracy
        train_preds = (preds >= 0.5).astype(int)
        train_acc = accuracy_score(y_train, train_preds)
        acc_history.append(train_acc)
```

```
    # Test Evaluation
    test_probs = sigmoid(np.dot(X_test_vec, W) + b)
    test_preds = (test_probs >= 0.55).astype(int)

    acc = accuracy_score(y_test, test_preds)
    prec = precision_score(y_test, test_preds)
    rec = recall_score(y_test, test_preds)
    f1 = f1_score(y_test, test_preds)

    return loss_history, acc_history, acc, prec, rec, f1

# ========================================
# 6. Try 5 Learning Rates
# ========================================
learning_rates = [0.001, 0.01, 0.05, 0.1, 0.2]
results = {}

# Proper figure creation
fig1, ax1 = plt.subplots()
fig2, ax2 = plt.subplots()

ax1.set_title("Loss vs Epochs (All Learning Rates)")
ax1.set_xlabel("Epoch")
ax1.set_ylabel("Loss")

ax2.set_title("Accuracy vs Epochs (All Learning Rates)")
ax2.set_xlabel("Epoch")
ax2.set_ylabel("Accuracy")

for lr in learning_rates:
    print(f"\n========== Learning Rate: {lr} ==========")

    loss_hist, acc_hist, acc, prec, rec, f1 = train_model(lr)

    results[lr] = f1

    print("Accuracy :", acc)
    print("Precision:", prec)
    print("Recall   :", rec)
    print("F1 Score :", f1)

    ax1.plot(loss_hist, label=f"LR={lr}")
    ax2.plot(acc_hist, label=f"LR={lr}")

# ========================================
# 7. Best Learning Rate
# ========================================
best_lr = max(results, key=results.get)
print("\n Best Learning Rate Based on F1:", best_lr)
```

```
# ========================================
# 8. Show Graphs
# ========================================
ax1.legend()
ax2.legend()

ax1.grid(True)
ax2.grid(True)

plt.show()


===== FIRST 5 ROWS =====
                                                email  label
0   date wed NUMBER aug NUMBER NUMBER NUMBER NUMB...      0
1  martin a posted tassos papadopoulos the greek ...      0
2  man threatens explosion in moscow thursday aug...      0
3  klez the virus that won t die already the most...      0
4   in adding cream to spaghetti carbonara which ...      0

===== LAST 5 ROWS =====
                                                email  label
2995   abc s good morning america ranks it the NUMBE...      1
2996   hyperlink hyperlink hyperlink let mortgage le...      1
2997   thank you for shopping with us gifts for all ...      1
2998   the famous ebay marketing e course learn to s...      1
2999   hello this is chinese traditional 子件NUMBER世...      1

========== Learning Rate: 0.001 ==========

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/
_classification.py:1565: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))

Accuracy : 0.8333333333333334
Precision: 0.0
Recall   : 0.0
F1 Score : 0.0

========== Learning Rate: 0.01 ==========

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/
_classification.py:1565: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

```
Accuracy : 0.8333333333333334
Precision: 0.0
Recall   : 0.0
F1 Score : 0.0

========== Learning Rate: 0.05 ==========

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/
_classification.py:1565: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))

Accuracy : 0.8333333333333334
Precision: 0.0
Recall   : 0.0
F1 Score : 0.0

========== Learning Rate: 0.1 ==========

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/
_classification.py:1565: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))

Accuracy : 0.8333333333333334
Precision: 0.0
Recall   : 0.0
F1 Score : 0.0

========== Learning Rate: 0.2 ==========

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/
_classification.py:1565: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))

Accuracy : 0.8333333333333334
Precision: 0.0
Recall   : 0.0
F1 Score : 0.0

 Best Learning Rate Based on F1: 0.001
```


Loss vs Epochs (All Learning Rates)
0.70
0.65
0.60
0.55
0.50
Loss
0
5
10
15
20
25
Epoch
LR=0.001
LR=0.01
LR=0.05
LR=0.1
LR=0.2


Accuracy vs Epochs (All Learning Rates)
Accuracy
Epoch
0.3
0.4
0.5
0.6
0.7
0.8
0
5
10
15
20
25
LR=0.001
LR=0.01
LR=0.05
LR=0.1
LR=0.2